CMS HCAL Test Beam Run Control: Post-Mortem Meeting

Ichiro Suzuki
CD/CEPA/OAA
Fermi National Accelerator Laboratory

HCAL run control, 2003/09/22

Post-Mortem Meeting

- Objective: Clarify problems in the project to improve the process of the future projects
- Outcome: List of things to do to prevent same mistakes
- Agenda
  - Introduction (5 min.)
  - Problems/Solutions, D.Charak (10 min.)
  - Problems/Solutions, I.Suzuki (15 min.)
  - Compilation of the action list
  - Meta-discussion on this meeting

CMS HCAL Beam Test '03

- CMS: an LHC experiment, starting in 2007
- HCAL: Hadron CALorimeter
- Beam test '03:
  - CERN H2 beam line
  - May '03: 25ns structured beam run with HB
  - Jul-Aug '03: normal beam run with HB, HF, HE and HO
- The DAQ system: (J. Mans, Princeton)
  - A Linux PC reads out a VME 9U crate, 50~1k events/bunch
  - Written with XDAQ (CMS online framework/toolkit)
  - MySQL based configuration and log DB
  - Slow control PCs connected via DIM

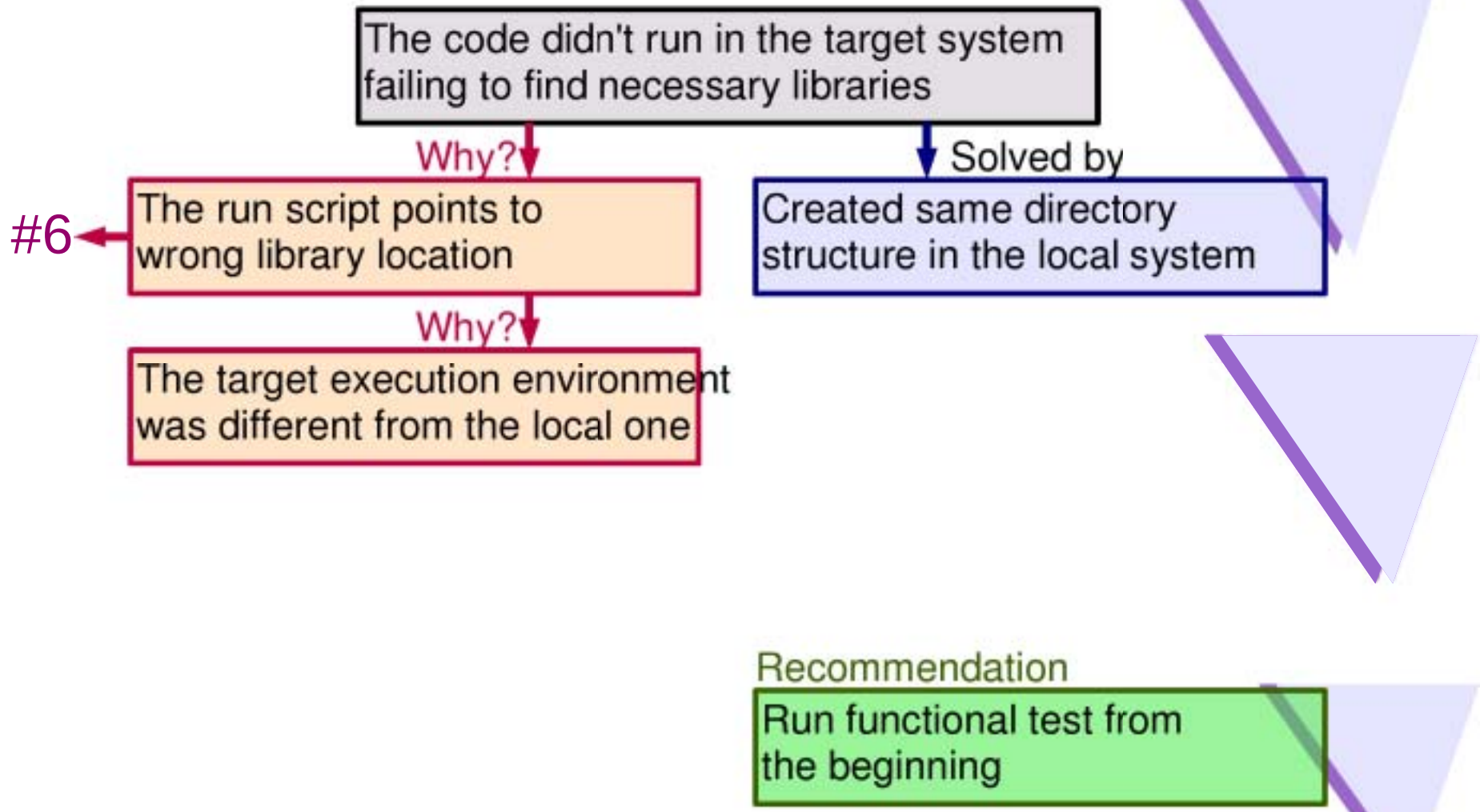
Run Control Development

- Started discussion at the beginning of May
- Object oriented development using Java
- Tried to learn the XP process
  - Test driven development
  - Pair programming
  - Simple design
  - Small releases
  - Continuous integration
  - Collective ownership

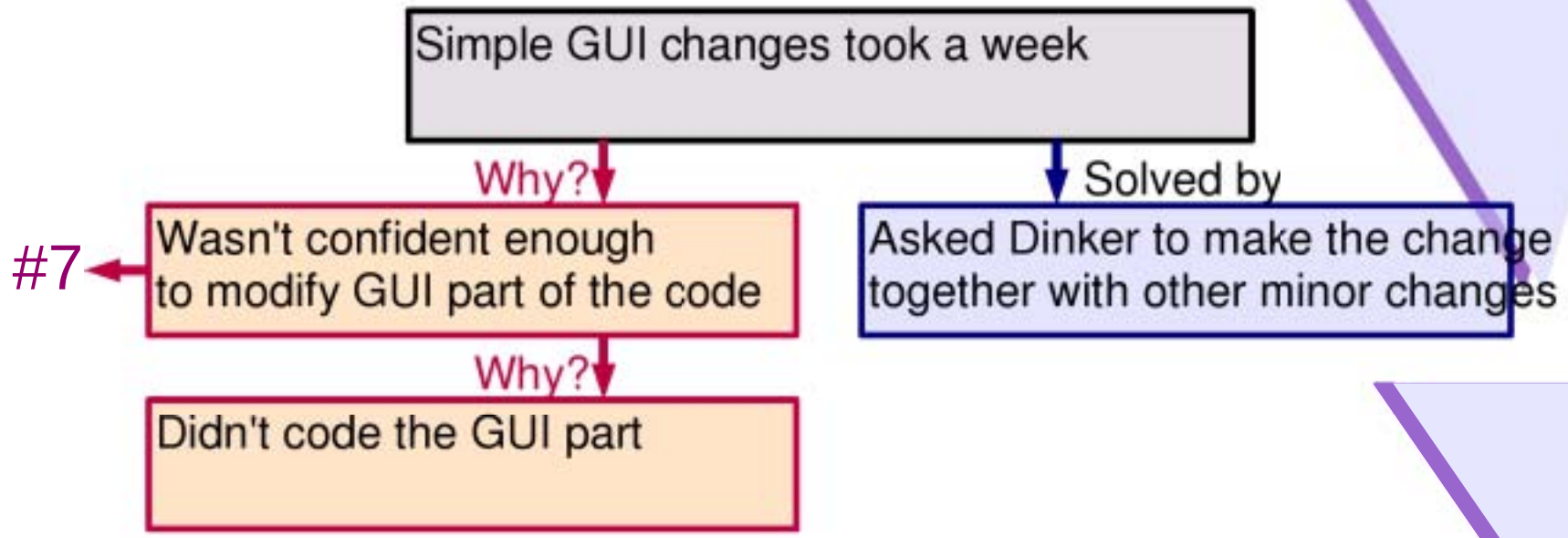
Status and Plan

- The run control was completed on time.
  - 2300 lines + 1300 lines of test code
- It has been used for all the data taking.
- It has been evolved as users made requests.
  - Automatic structural XML file retrieval from the DB
  - Sequencer mode for beam scan (moving table) and
    source scan (moving source driver) for the HF
- It was a successful project!

#1


The code didn't run in the target system failing to find necessary libraries
Why?
Solved by
#6
The run script points to wrong library location
Created same directory structure in the local system
Why?
The target execution environment was different from the local one
Recommendation
Run functional test from the beginning

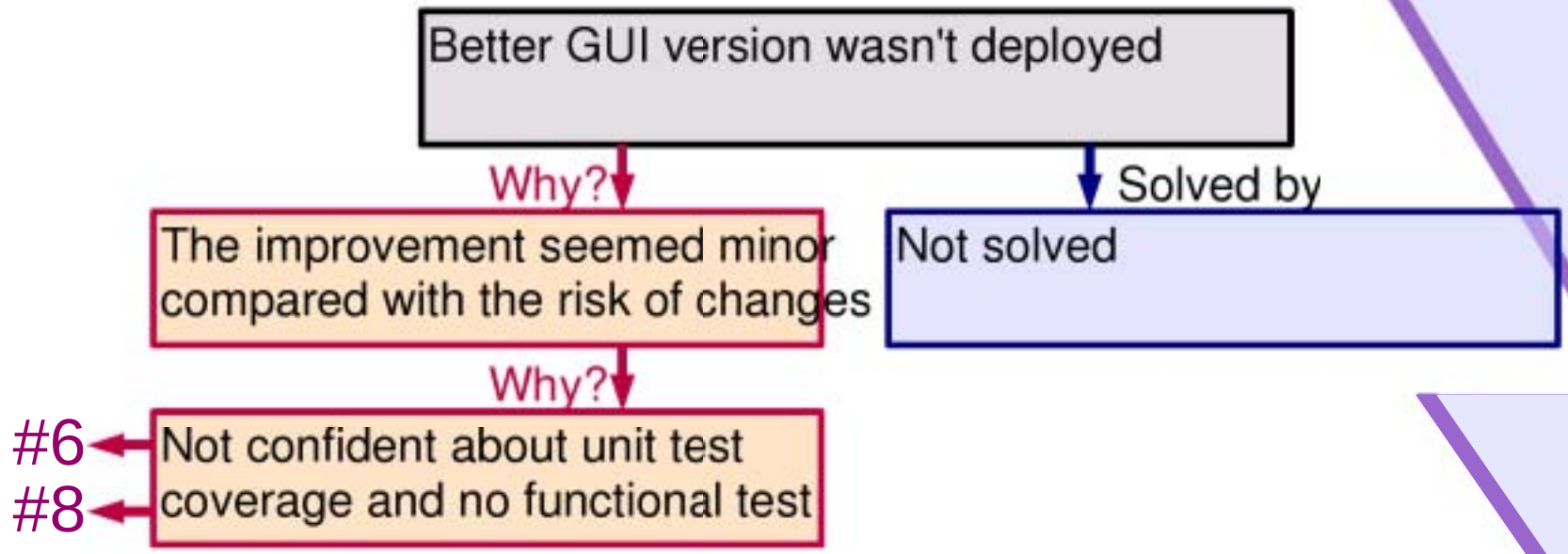
#2



Recommendation

Pair programming to share knowledge of the code

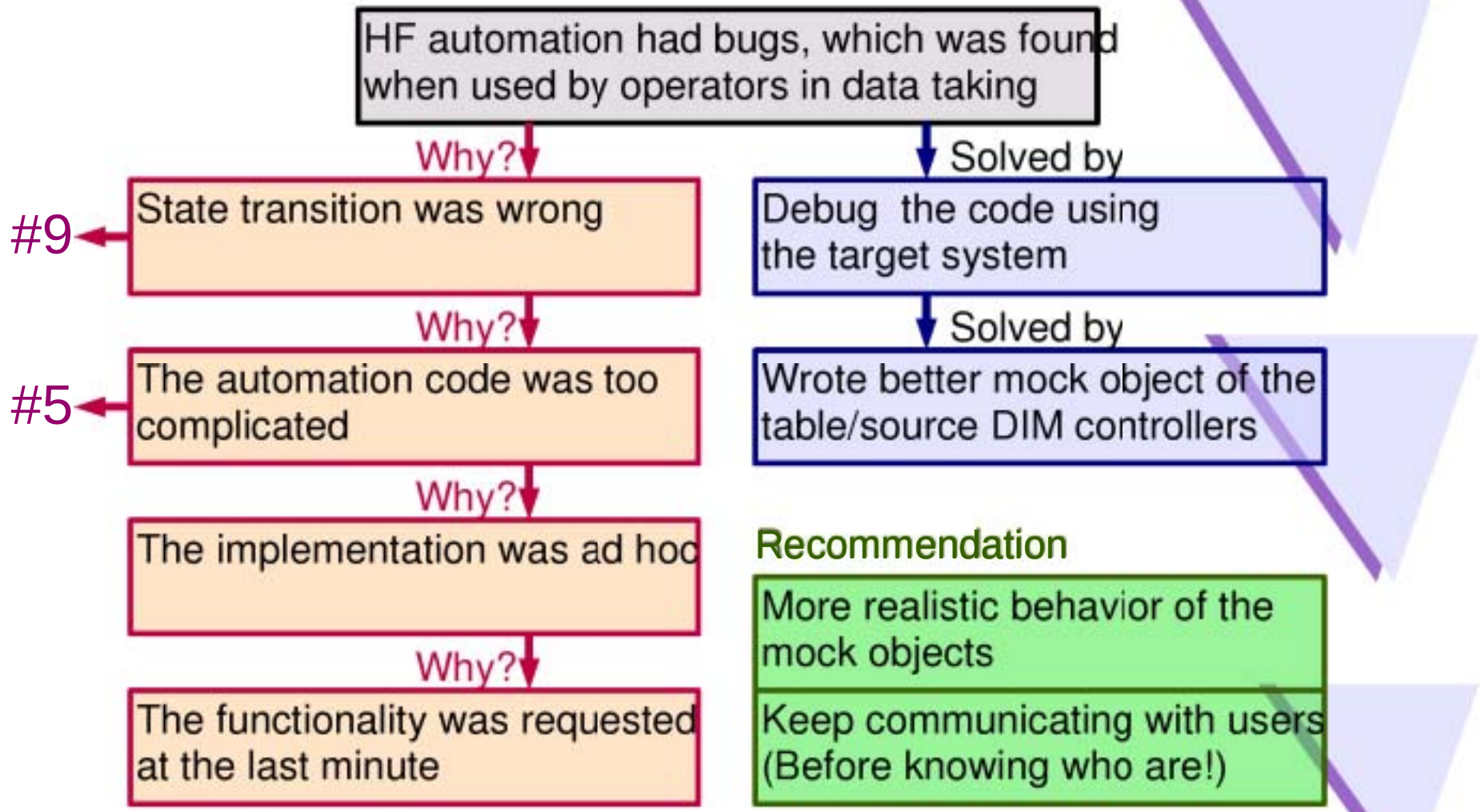
#3



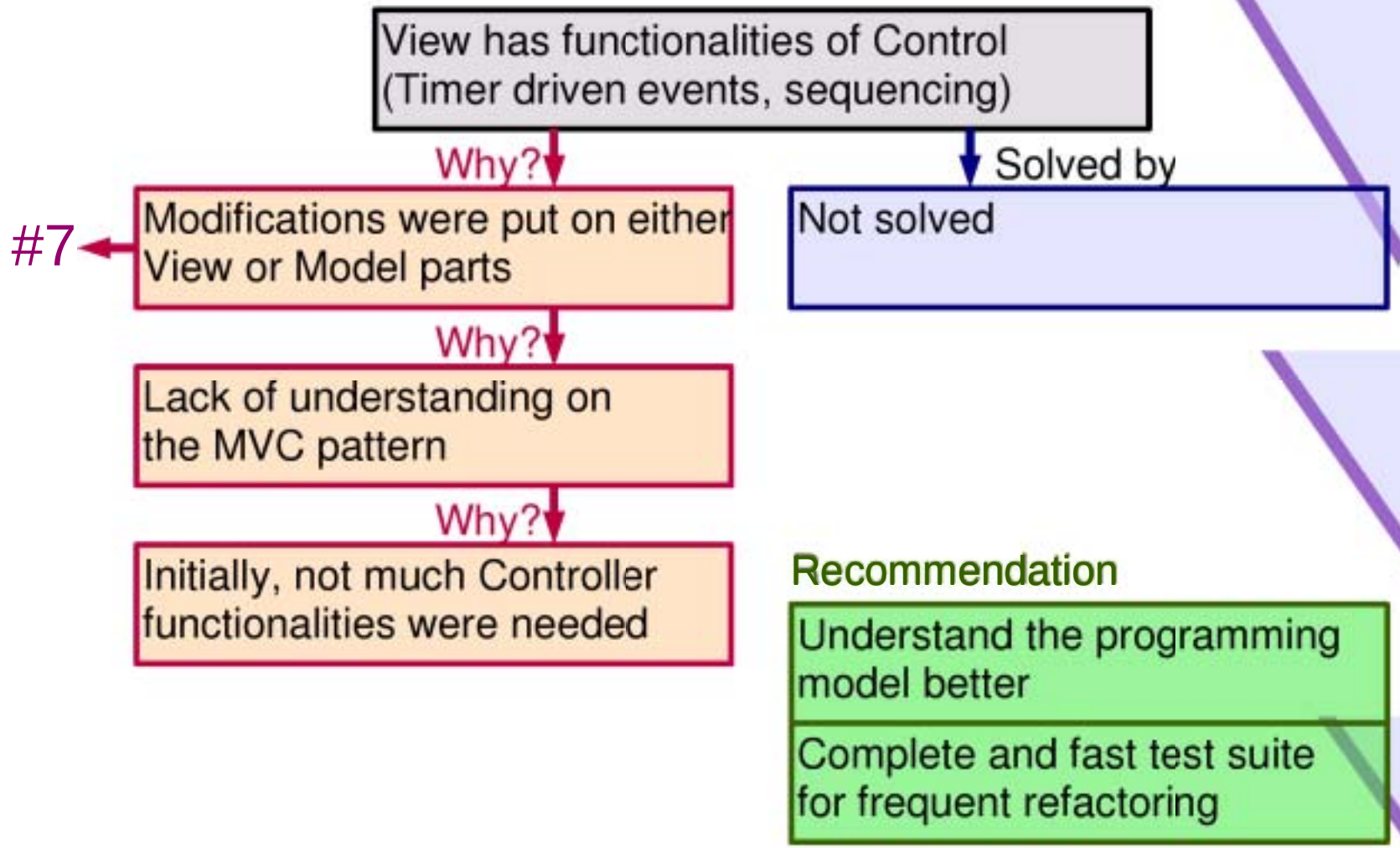
Recommendation

Comprehensive unit tests and functional tests

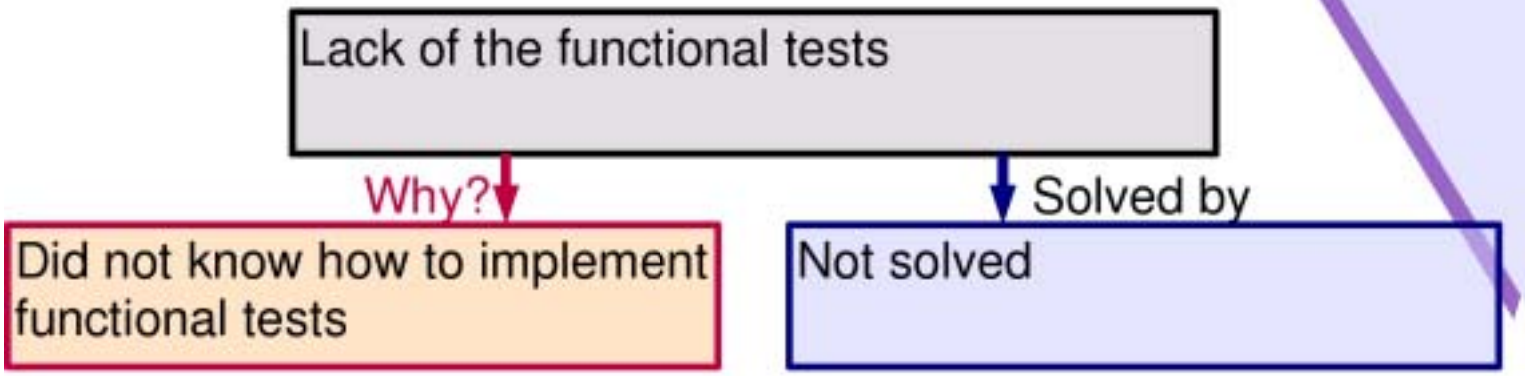
#4


HF automation had bugs, which was found when used by operators in data taking
Why?
Solved by
#9
State transition was wrong
Debug the code using the target system
Why?
Solved by
#5
The automation code was too complicated
Wrote better mock object of the table/source DIM controllers
Why?
The implementation was ad hoc
Recommendation
More realistic behavior of the mock objects
Why?
The functionality was requested at the last minute
Keep communicating with users (Before knowing who are!)

#5


View has functionalities of Control
(Timer driven events, sequencing)
Why?
Solved by
#7
Modifications were put on either
View or Model parts
Not solved
Why?
Lack of understanding on
the MVC pattern
Why?
Initially, not much Controller
functionalities were needed
Recommendation
Understand the programming
model better
Complete and fast test suite
for frequent refactoring

#6

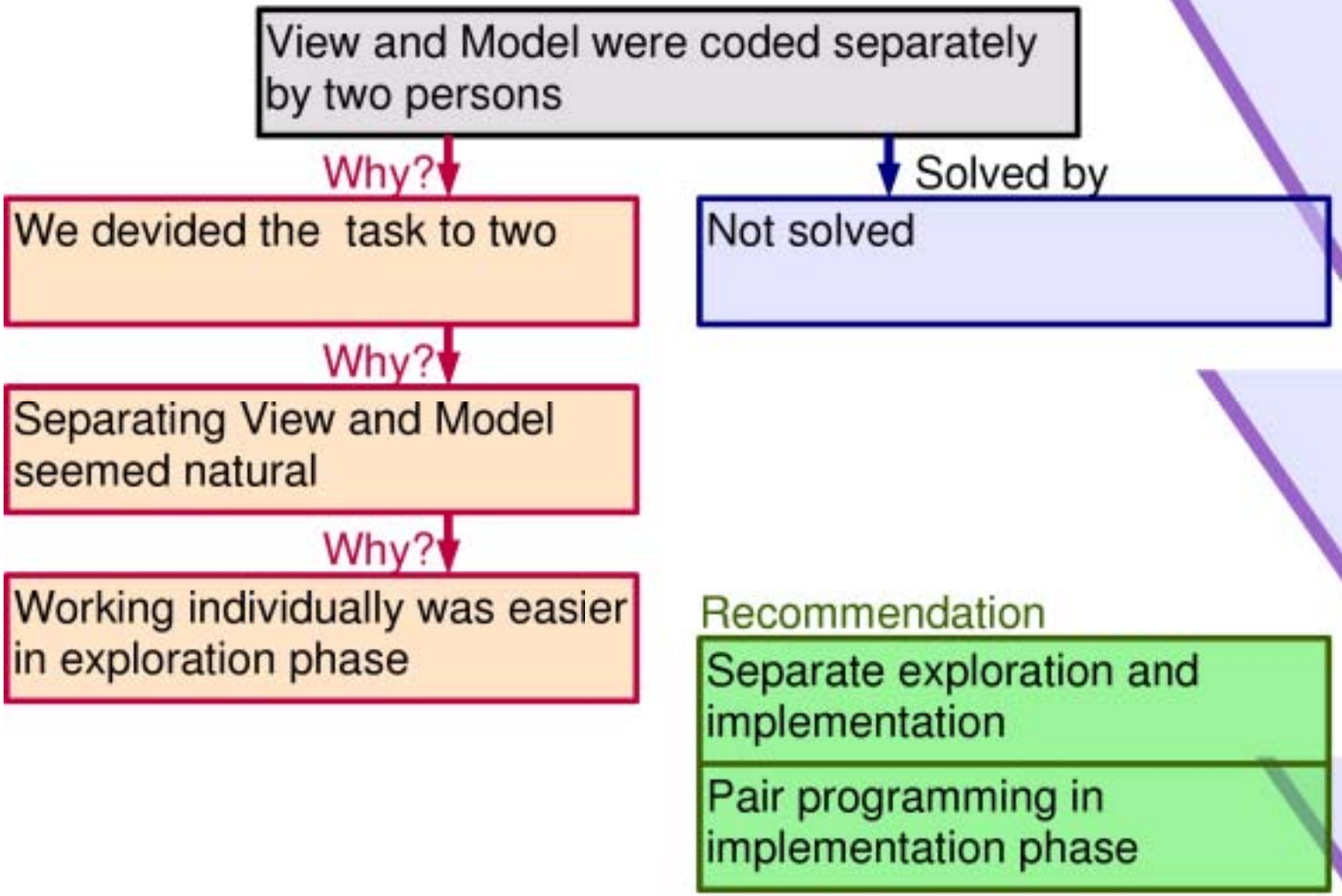


Recommendation

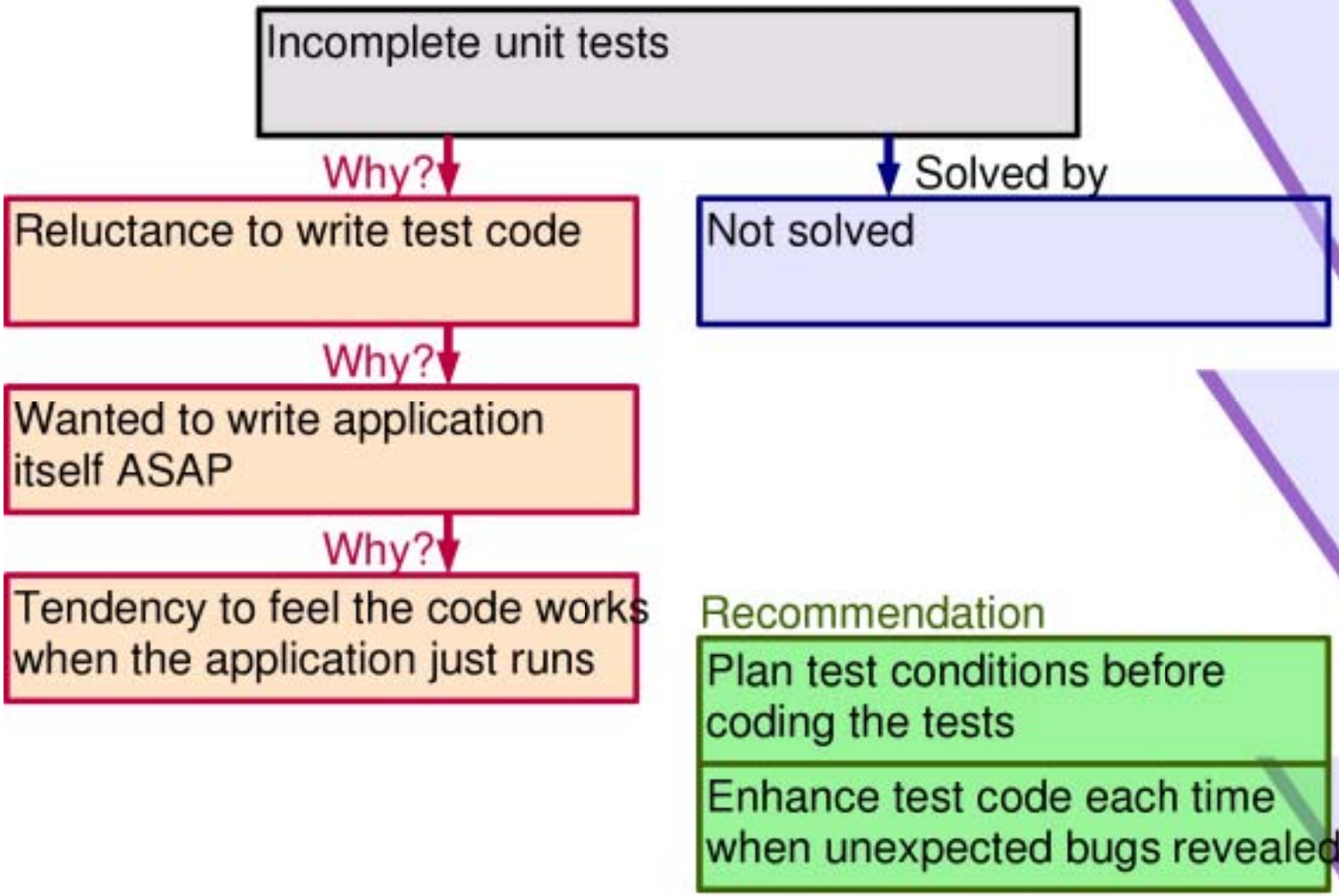
'Planning game' to identify what should be tested

Write functional tests

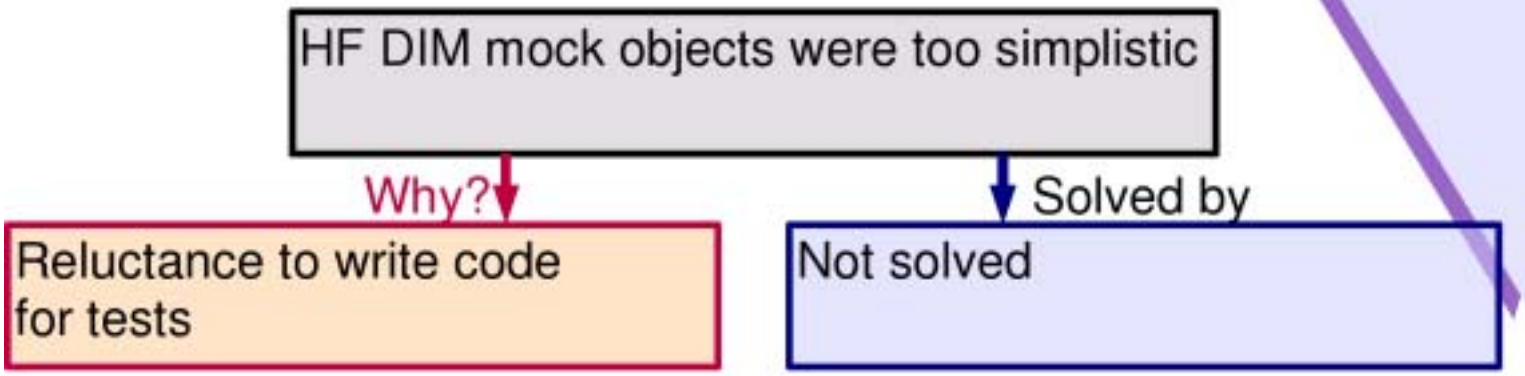
#7


View and Model were coded separately by two persons
Why?
We devided the task to two
Why?
Separating View and Model seemed natural
Why?
Working individually was easier in exploration phase
Solved by
Not solved
Recommendation
Separate exploration and implementation
Pair programming in implementation phase

#8


Incomplete unit tests
Why?
Solved by
Reluctance to write test code
Not solved
Why?
Wanted to write application itself ASAP
Why?
Tendency to feel the code works when the application just runs
Recommendation
Plan test conditions before coding the tests
Enhance test code each time when unexpected bugs revealed

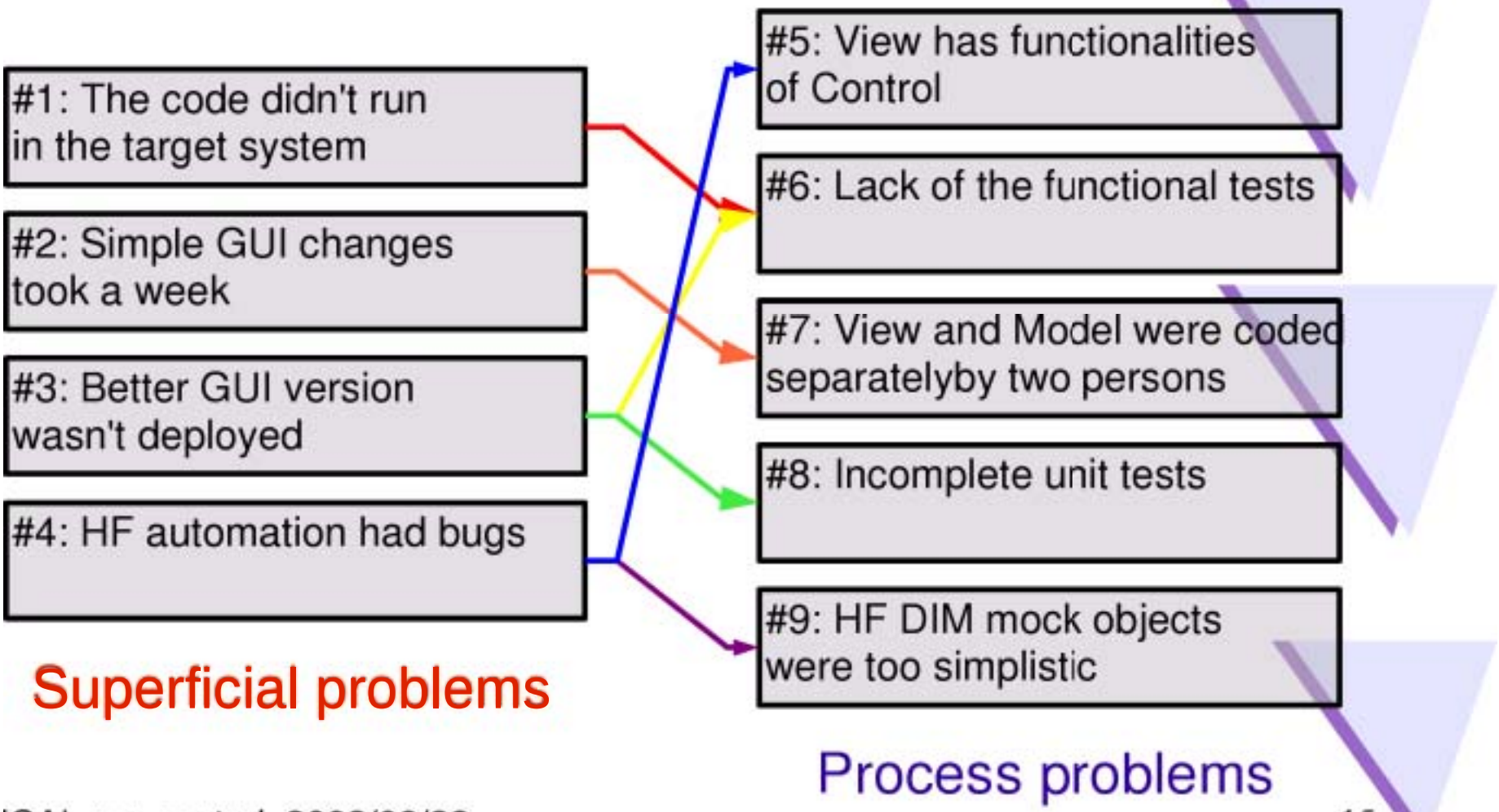
#9



Recommendation

- Don't try to save time to write test code
- Realistic enough mock objects

Relations of the Problems



List of Recommendations

- Reinforce XP practices
  - Planning game
  - Test first (functional + unit tests)
  - Pair programming
  - Refactoring
- Find out users requests by close communication
- Understand design patterns
- Put efforts on the test code
  - Environment
  - Mock objects
  - Variety of test conditions